

Automated Multi-Line Labeling of a Time Axis**By Lory Molesky****Field of the Invention**

5 The present invention relates to a method of labeling the axes of graphs and charts.

Background of the Invention

10 Effectively conveying time-based data often requires creating charts and graphs that are easily understood and appreciated. In today's business world, a superior presentation of data may greatly impact a business's performance. The primary purpose of a graph is to quickly communicate information to an intended audience. An effective graph conveys information to a viewer in a clear, well-designed and unambiguous manner. Thus, an effective graph has meaningful labels,
15 clear axis marks, organization, and is uncluttered. The success of graph often depends upon whether the graph and its labels are aesthetically pleasing.

 Graphs and charts are used in a variety of fields, such as finance, e-commerce, communications, consulting, accounting, law, marketing, sales, etc. More and more these businesses are having to deal with a large influx of data and they must find a
20 way to quickly and effectively communicate the data. As data becomes more available businesses need an automated process to create comprehensible graphs and charts. This need is most acute when handling time based data because, time units

have unusual variations and particularities. Thus, an automated process that converts time based data into an elegantly labeled graph is needed.

The automated labeling of a time axis is one of the most challenging components of any charting software. Commercially available charting software does not automate the labeling of the time axis in a way that is visually pleasing. For example, many spread sheet packages only use a single line of text to label the time axis. The resulting chart is cluttered and has redundant labels. In contrast, leading business publications employ multi-line labeling of the time axis in their presentation graphics. However, these labels are not automated and require human operators (e.g. graphic artists) to perform the layout manually. Thus, the general problem is finding a method to elegantly and automatically label a chart or a graph with a time axis for any time unit.

Summary of the Invention

The present invention is an automated method and system of labeling of the axes of a graph. The present invention dynamically labels a graph axis with time based data. The labels on the axis adjust for the amount and type of data provided. The labels have multiple levels so that graph appears uncluttered and adjust for a display's size.

The invention generates a set of time labels by processing the input data; the time labels are stored in a multi-level data structure. The multi-level data structure is then processed to refine the time labels. The refining step includes adjusting the labels

so that they fit in the space provided along the axis. The time labels that are stored in the multi-level data structure are then used to generate multi-level time labels. These multi-level time labels are used to label the axis of the graph. The invention automatically provides an effective, elegant, labeled time based graph.

5 One of the advantages of the current invention is that the software creates a dynamically labeled time axis that adjusts for the display size and for the amount of data that is provided. Further, the present invention automatically generates an elegant layout based on the data provided, the available real estate (i.e. the length of the time axis) and the desired font.

10 Another advantage of the invention is that the method of labeling may be applied to any type of charting software. The process is generic and is not software specific.

Another advantage of the present invention is that knowledge of time units and their peculiarities is built into the system thereby enabling "known" time labels.

15 Another advantage of the invention is that the present invention is independent of any display device and, hence it can be utilized for any type display device.

Yet another advantage of the invention is that knowledge of the displays may be built into the system, enabling automatic labeling and gridding.

20 Yet another advantage of the present invention is that is produces non-cluttered multi-level labels for a graph.

Brief Description of the Drawings

The details of the present invention, both as to its structure and operation can best be understood by referring to the accompanying drawings.

Figure 1 is an environmental diagram showing the data flow in the present invention.

Figure 2 is an example of a graph with a labeled time axis shown on a display device.

Figure 3 provides various sample graphs may be outputted by the present invention.

Figure 4 is a flow diagram of the automated labeling process.

Figure 5 is a flow diagram showing the details of generating the time labels.

Figures 6a and 6b are exemplary diagrams of multi-level data structures.

Detailed Description of the Invention

The present invention is a method and system for automatically labeling an axis of a graph with time based data. The data flow of the present invention is provided in figure 1. Input data 2 is provided to a system implementing the time axis labeling 4. The input data 2 is processed by the time axis labeling method 4 and a labeled graph is outputted on a display device 6. The display device 6 may be the screen of a computer 8, a printer, or the like. An exemplary diagram of a display device 6 showing a labeled graph is provided in figure 2. The labeled graph has a time

axis 10 and multi-level time labels 12. The labeled graph also has an informational data axis 14.

As provided in figure 1, input data 2 includes the length of the time axis 10, the size and type of font used to label the time axis 10, informational data and corresponding timestamps, and time units of the inputted timestamps. The time units, such as days, minutes or seconds, may be defined by the user or selected from a time dimension hierarchy. A time dimension hierarchy is a predefined array of time units. One example of a time dimension hierarchy is "Year, Month, Day, Minute, Second". Another example of a time dimension hierarchy is "Year, Month, Week, Day". Any type of time unit may be included in the time dimension hierarchy such as milliseconds or $\frac{1}{2}$ days.

The output of the software program is a labeled graph such as the one shown in figure 2. Further examples of outputted labeled graphs are shown in figure 3. As provided in these figures, typically, the x-axis of the graph is the time axis 10 and the y-axis is the informational data axis 14. Each of the graphs has multi-level time labels 12 on the time axis 10. Multi-level time labels 12 are composed of time based data and have multiple levels or rows. Multi-level time labels often contain abbreviated time units. Further, another language for the multi-level time labels may be utilized. As seen in figure 3, the multi-level time labels vary with the amount, type and variation of the input data. In other words, the format of the multi-level time labels is driven by the input data. Thus, the layout, format, and spacing of the multi-level time labels are customized to best label a chart with the input data provided.

Overall, the method of automatically labeling a time axis of a graph with time based data includes six steps. A flow diagram of the steps is provided in figure 4.

First, a set of time labels is generated 16 by processing the input data 2. Second, a multi-level data structure is created 18. Third, the time labels are stored in the multi-

5 level data structure 20. Fourth, the multi-level data structure is then processed to refine the time labels 22. Fifth, multi-level time labels are generated from the time labels stored in the multi-level data structure 24 sixth, the multi-level time labels are used to label the axis of the graph 26. Finally, the labeled graph is outputted to a display device. The present invention may be implemented with software or

10 hardware. A software programming language such as Java is appropriate.

Step 16 describes generating a set of time labels. A flow diagram of the process that generates a set of time labels is provided in figure 5. The input parameters (length of the time axis, size and type of font, the timestamps, and the time units) are examined in this step. First, an initial set of time labels is created by

15 reviewing the inputted timestamps and time units 29. Starting with the least varying time unit of the timestamps (such as YEAR), and ending with the most varying time unit of the timestamps (such as SECOND), each of the time unit hierarchies is traversed. Each level of the time dimension hierarchy is reviewed and a set of time labels is generated to correspond with that level. Those labels are then checked to see

20 if they fit along the time axis 30. If they fit, then the next level of the time dimension hierarchy is reviewed. The initial set of time labels includes the labels for all dimensions which fit along the time axis. If the set of time labels do not fit, then an

abbreviated set of time labels is created 34. The abbreviated set of time labels is then checked to determine if the set will fit along the time axis without overlapping 30. If the abbreviated set of time labels fits then that set is included as one of the time label sets 37. The set of time labels is saved and the next step of the method is to proceed to the “current” time dimension hierarchy level 30. If the set of time labels does not fit, then a subset of time label is created 36. The subset of time labels is then checked to determine if the set will fit along the time axis without overlapping 30. If the subset of time labels fits then that set is included as one of the time label sets 37. The set of time labels is saved and the next step of the method is to proceed to the “current” time dimension hierarchy level 42 and check to see if the the set of time labels fit along the time axis without overlapping 30. If the set of time labels do not fit, then the method ceases 48.

Step 30 accepts the initial set of time labels and determines whether the initial time labels can be written in the allotted space on the time axis 10. The lengths of each time label and an appropriate inter-label spacing constant is summed. The inter-label spacing constant is a predetermined minimum distance between each time label, the inter-label spacing constant may vary depending upon the font utilized. The sum is compared with the length of the time axis 10. If step 30 determines that the initial time labels will fit in the allotted space on the time axis 10 then those time labels will be used and the next step 18, referred to in figure 4, is performed. If step 30 determines that the time labels will not fit on the time axis 10 without overlapping then, the program creates an abbreviated set 34 of those time labels.

Step 34 generates an abbreviated version of the time labels. Abbreviations may be predefined or user defined. In addition, for the year unit, label prefixes may be factored (e.g. 1995 '96 '97 '98 '99 2000). The abbreviated version is an adaptation of the original time labels ; the abbreviated version contains all of the original time labels in abbreviated form. For example, if the unit name is supposed to display year and month ("full year" over "full month"), two abbreviated versions may be generated. One abbreviated version may be the "two digit year" over the "abbreviated month" and the other abbreviated version may be "full year" over several "single character months". Each of these abbreviated versions is tested in step 30 to determine if the abbreviated version of the time labels will fit along the time axis 10. If either abbreviated version fits then those abbreviated time labels will be used and the next step 18, referred to in figure 4, is performed. If neither abbreviated version fits on the time axis, then the program creates a time label subset 36.

Step 36 generates a subset of the time labels. For example, assume the time labels were every year from 1975 to 2000. Ever fifth label may be selected yielding the subset 1975 1980 1985 1990 1995 2000; this would be a subset of time labels. A label constant may be considered by the program to produce a more recognizable subset. The label constant determines the frequency of the time label. The frequency is the amount of time a label occurs. The label constant guides the software program in choosing a more comprehensible subset. The label constant is dependent on the time unit and may vary for each time unit. For example, consider the situation where the step 36 generates a subset containing every 13th label. But, every 13th label is not

easily recognized. If the label constant is used and is set at 5, every 15th label is chosen because it is the next frequency of label that is divisible by the label constant.

Also, a threshold may be used in step 36. For example, assume the threshold is set at 30 and the label constant is set at 5. The original set of data (11 to 45 seconds)

5 translates 11 to 14 seconds => 15 seconds, 16 seconds to 19 seconds => 20 seconds, 21 seconds to 29 seconds => to 30 seconds. All periods greater than 30 seconds are rounded to a minute. Also, a mapping function may be used that will map certain data or data ranges to other values. Thus, label constants, thresholds, and mapping functions assist in generating a set of labels that is standard and identifiable.

10 Once a subset of time labels is generated, the program initiates step 30 to determine if the subset fits along the time axis. If the subset fits, then that subset of time labels is used and the next step 18, referred to in figure 4, is performed. If the subset fails step 30, an abbreviated version of the subset is generated. The process is figure 5 is repeated until a set of time labels 14 is generated that will fit along the time
15 axis 10 without overlapping. Then those time labels will be used and the next step 18, referred to in figure 4, is performed.

Step 18 is creating a multi-level data structure 38. Pictorial examples of multi-level data structures 38 are shown in figures 6a and 6b. The multi-level data structure 38 may be viewed as an array. The multi-level data structure 38 has areas to store
20 time labels as character strings. The multi-level data structure 38 has an internal index system that is independent of any display device 6. This index system allows the formatting of the time labels to be compatible with any display device. The index is a

set of coordinates for each time label. Later in the process, the indexes are translated to a display's system so that the graph can be properly displayed.

Step 20 is storing the time labels in the multi-level data structure 28. Time labels are stored in a multi-level data structure 38 with one time unit per level. The multi-level data structure 38 is populated by reviewing the frequency of the time units of the labels. The frequency is the amount of time a unit occurs in a given period. For example, seconds have a higher frequency than minutes and minutes has a higher frequency than hours. The time labels with the highest unit frequency are stored at the "logical" top of the multi-level data structure 38. Then, in descending order of unit frequency, the time labels stored on each level of the multi-level data structure 38. If there is no difference in the frequency of the time units, the order of time labels is not set (for example, numeric days and character days). For example, referring to figure 6a, hours are stored on the top level L0, days are stored on a middle level L1, numeric days are stored on the next level L2, months are stored on level L3 and years are stored on the bottom level L4. Each level contains one or more time labels but only one time unit. The multi-level data structure 38 also stores the layout of the time labels.

While storing the time labels in the multi-level data structure 38, each time label is indexed. The time label has a start and an end index. This is demonstrated in the following example. Assume the time labels are 01-Nov-1999 01-Dec-1999 01-Jan-2000 01-Feb-2000 01-Mar-2000. The multi-level data structure would be:

Level 0: Nov[0,1], Dec [1,2], Jan[2,3] Feb [3,4], Mar [4,5]

Level 1: 1999 [0,2], 2000 [2,5]

The indexes provide the information necessary to align the time labels relative to each other. For example, 1999 [0,2] is located on level 1, the label character string is 1999, the starting index is 0 and the ending index is 2. This means that the label “1999” occupies the interval on level 1 that corresponds with data spaces 0 to 2. This is a device independent representation. An example of a section of a Java software program is provided to more clearly illustrate how this step may be implemented. The following code sets forth the character string (i.e. the actual time label), and the starting index.

```
10      Class label      {  
        String labelString; // The actual label to write; the character string  
        int start_index;    // start index of interval (of Tstamp[])  
        int end_index;      // end index of interval (of Tstamp[])  
      }
```

15 The next portion of code defines a vector that corresponds to one level of the multi-level data structure 38. This data structure is a vector of labels.

```
      Private class Linelabel    {  
        Vector Vlabel = new Vector ();  
      }
```

20 Finally the vector/level is assigned a particular time unit. The data structure is a vector of linelabel.

```
      Public class Xlabel {
```

Vector Xlabel Unit = new Vector();

Public class time labels {

Vector Time Label Levels = new Vector ()

5 Step 22 is processing the multi-level data structure 38 to refine the time labels stored in the multi-level data structure 38. Step 22 extends the precision of the time labels, rewrites time labels, and merges levels in the multi-level data structure 38. The purpose of this step is to refine the appearance and organization of the time labels.

The multi-level data structure 38 is reviewed to determine if any of the time labels may be made more precise. For example, figure 6a has a time label on level L0 that is "5pm." This time label may be rewritten, as shown in figure 6b, to be "5:30pm." Typically, the level containing the highest frequency time unit, here L0, is rewritten with extended time precision. In particular, the initial time label and the last time label may be rewritten to make these time labels more precise. Another example of this follows - suppose the first time label started at 9:30, but time labels were generated only for whole hours. This occurred in step 16 because it was determined that there was not enough space to label minutes and there was enough space to label hours. To make the graph more accurate and intuitive, the initial time stamp may be rewritten . So, in this example the following transformation may take place if 9:30 will fit into the allotted space.

9 10 11 12 ... => 9:30 10 11 12 ...

The transformation is important because it more accurately displays the precision of the initial timestamp.

Next, invariant time labels may be merged. An invariant time label is a label where there is only one label in a level; examples are provided below. There may be levels that contain only one time label. These are termed invariant time labels, and are merged into other levels. Examples of invariant time labels are shown in figure 6a. L3 Nov.[0,8] and L4 2000 [0,8] are invariant time labels. The first situation where invariant time labels are merged is where all the time labels in the multi-level data structure are invariant time labels. In this case, provided there is enough space, all invariant time labels are merged. For example, if the input was daily timestamps for the month of September 1999, and there was not enough space to write daily time labels, the following transformation would occur:

Sept. => Sept. 1999
1999

The second situation where invariant time labels are merged is where the levels contain both invariant and variant time labels. This is the most general case. The overall strategy is to merge all invariant time labels. For example, the program may make the following transformation so that the time labels are clearer:

9am 10am 11am 12am 9am 10am 11am 12am
2 => September 2, 1999
September
1999

It may be the case that an invariant time label is merged into a single line that contains variant time labels. For example, given that there is enough space for the 2000 to fit after Jan, the following transformation will take place:

5 Jan Feb Mar Apr => Jan 2000 Feb Mar Apr
 2000

Step 24 is generating multi-level time labels 12 from the multi-level data structure 38. Here, the time labels are structurally aggregated. In another words, instead of having individual time labels, the time labels are aggregated with those in the “column” of the multi-level data structure. Thus, a multi-level time label 12 is
10 formed instead of individual time labels.

Step 26 is labeling the graph with the multi-level time labels 12. The multi-level time labels 12 are rendered onto the time axis of the graph. The graph is then displayed. Display devices 6, such as a computer screens, are typically addressed in terms of device display coordinates or pixel coordinates. These device display
15 coordinates indicate where a multi-level time label 12 will be displayed in the image. Functions are used to convert the indexes into device display coordinates. The functions support date arithmetic and provide an interface for extracting time fields, such as year, month, day, hour, etc., from the input data 2. Further, the functions may also convert numerical data to character strings such as “11” to “November”. Finally,
20 the chart or graph is produced with an elegantly labeled time axis 10. Examples of outputted graphs are shown in figure 3.

Axis markers may also be defined. Axis markers (also known as hash marks) are used to define increments along the time axis that are too small to be labeled. Axis markers may be written for non-labeled time units; axis markers will be written for the lowest frequency non-labeled time unit. Axis markers are defined in a similar fashion to time labels. A vector of axis marker is defined; each axis markers contains indexes that indicate where the axis marker should be written. The indexes are not display device specific. For example, a code could be written in Java such as the one that follows:

```
Class HashMarks {  
    Vector Vmarks = new Vector();  
    Int period = 1;  
}  
Class Mark {  
    Vector Vmark = new Vector ();  
    Int period = 1;  
}  
Class Mark {int index}
```

It is important to note that while the present invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in the form of a computer readable medium of instructions and a variety of forms and that the present invention applies equally regardless of the particular type of

signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media such as floppy disc, a hard disk drive, RAM, and CD-ROM's, as well as transmission-type media, such as digital and analog communications links.

- 5 Although specific embodiments of the present invention have been described, it will be understood by those of skill in the art that there are other embodiments that are equivalent to the described embodiments. Accordingly it is to be understood that the invention is not to be limited by the specific illustrated embodiments, but only be the scope of the appended claims.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100